Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Jernej Muha

Razvoj večplatformne hibridne mobilne aplikacije za iskanje ponudnikov študentske prehrane

DIPLOMSKO DELO

UNIVERZITETNI ŠTUDIJSKI PROGRAM PRVE STOPNJE
RAČUNALNIŠTVO IN INFORMATIKA

Ljubljana, 2016

Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Jernej Muha

Razvoj večplatformne hibridne mobilne aplikacije za iskanje ponudnikov študentske prehrane

diplomsko delo

univerzitetni študijski program prve stopnje
računalništvo in informatika

Mentor: doc. dr. Aleš Smrdel

Ljubljana, 2016

Besedilo je oblikovano z urejevalnikom besedil LaTeX.

Fakulteta za računalništvo in informatiko izdaja naslednjo nalogo: Razvoj večplatformne hibridne mobilne aplikacije za iskanje ponudnikov študentske prehrane

Tematika naloge:

Zgodi se, da pridejo študenti v mesto ali predel mesta, ki ga ne poznajo, radi pa bi tam jedli v lokalu, ki ponuja subvencionirano prehrano. V takih primerih bi zelo prav prišla mobilna aplikacija, ki bi uporabniku prikazala bližnje lokale, ki ponujajo subvencionirano prehrano. V okviru diplome razvijte mobilno aplikacijo, ki bo ugotovila lokacijo uporabnika preko sistema GPS mobilne naprave. Nato se bo povezala z oddaljenim strežnikom in pridobila podatke o lokalih, ki ponujajo subvencionirano prehrano. Na podlagi pridobljenih informacij pa bo mobilna aplikacija na zemljevidu prikazala lokale, ki so v bližini. Aplikacija naj omogoča tudi podroben prikaz informacije o izbranem lokalu. Aplikacijo razvijte vsaj za eno mobilno platformo. Razvijte tudi strežniški del, ki bo zagotavljal potrebne podatke za mobilno aplikacijo. Razvito mobilno aplikacijo poskusite tudi objaviti v trgovini za izbrano platformo. Na podlagi zahtevkov, ki bodo poslani strežniku, pa v diplomi tudi prikažite statistiko uporabe razvite aplikacije.

Hvala očetu in materi, ki sta mi omogočila študij ter mentorju doc. dr. Alešu Smrdelu za pomoč in nasvete pri izdelavi tega diplomskega dela.

Kazalo

Seznam uporabljenih kratic

kratica	angleško	slovensko
HTML	Hyper text markup language	jezik za označevanje nadbesedila
CSS	Cascading style sheets	kaskadne slogovne podloge
SDK	Software development kit	orodja za razvoj programske opreme
API	Application programming interface	aplikacijski programski vmesnik
I/O	Input/output	vhod/izhod
CLI	Command-line interface	vmesnik preko ukazne vrstice
REST	Representational state transfer	predstavitveni prenos stanja
PaaS	Platform as a service	platforma kot storitev
MVC	Model-View-Controller	model-pogled-kontroler
JSON	JavaScript object notation	objektna notacija JavaScript

Povzetek

Naslov: Razvoj večplatformne hibridne mobilne aplikacije za iskanje ponudnikov študentske prehrane

Cilj diplomske naloge je bil razviti večplatformno hibridno mobilno aplikacijo za iskanje ponudnikov študentske prehrane. Glavna funkcionalnost aplikacije je hiter in enostaven prikaz lokacij najbližjih restavracij s študentsko prehrano glede na uporabnikovo trenutno lokacijo. Hkrati z opisom celotnega razvoja hibridne mobilne aplikacije so opisane tudi tri različne vrste razvoja za mobilne naprave in razlike med njimi ter prednosti in slabosti posameznih vrst mobilnih aplikacij. Predstavljen je tudi izgled in obnašanje aplikacije ter analiza njene uporabe. Za razvoj odjemalca je bilo uporabljeno orodje Ionic, ki nadgrajuje tehnologiji AngularJS za gradnjo spletne aplikacije in Apache Cordova za pretvorbo le te v izvirno mobilno aplikacijo. Srednji sloj aplikacije je bil razvit v okolju Node.js z uporabo paketa LoopBack. Za podatkovno bazo je bil izbran NoSQL ponudnik MongoDB, za njeno gostovanje (in gostovanje srednjega sloja) pa oblačna storitev Heroku. Za analizo uporabe aplikacije je uporabljena oblačna storitev Ionic Analytics.

Ključne besede: hibridna mobilna aplikacija, študentska prehrana, Ionic, AngularJS, Apache Cordova, LoopBack, MongoDB, Heroku.

Abstract

Title: Development of cross-platform hybrid mobile application for finding providers of student meals

The aim of the thesis was to develop a cross-platform hybrid mobile application for searching providers of student meals. The main functionality of the application is a quick and easy view of the nearest locations of restaurants with student food, depending on the user's current location. While describing the whole development of hybrid mobile applications, we are also describing three different types of development for mobile devices, the differences between them and the advantages and disadvantages of different types of mobile applications. Also the appearance and behaviour of the application and analysis of its usage usage is presented. Ionic framework has been used for client development, which is built on top of AngularJS for building web applications and Apache Cordova for creating hybrid mobile applications. The middle-tier has been developed in Node.js environment using LoopBack package. MongoDB was used for a database provider and Heroku for its hosting (also hosting the middle-tier). Ionic Analytics is used for the analysis of application usage.

Keywords: hybrid mobile application, student meals, Ionic, AngularJS, Apache Cordova, LoopBack, MongoDB, Heroku.

Poglavje 1

Uvod

Le nekaj let nazaj se je razpravljalo, ali je razvoj mobilne aplikacije vreden časa in stroškov, danes pa je pomembnost mobilnosti in aplikacij na mobilnih platformah več kot očitna. Tako je na primer od junija 2016 na voljo več kot štiri miljone aplikacij v trgovinah Google Play in Apple App Store [6]. V letu 2015 je bilo prodanih šest krat več pametnih mobilnih telefonov kot namiznih in prenosnih računalnikov [3]. Do leta 2017 se pričakuje, da bo število tabličnih računalnikov večje od števila prenosnih in namiznih računalnikov [3]. Samo na podlagi predstavljenih dejstev lahko rečemo, da je mobilnost postala del življenja.

Leta 2010 je bil razvoj aplikacij za mobilne naprave osredotočen v razvoj izvirnih (*ang. native*) aplikacij. Pri razvoju izvirnih mobilnih aplikacij se uporabljajo različni jeziki za različne platforme, kot so na primer Java za platformo Android, objektni C za platformo iOS in C# za platformo Windows Phone. Poleg tega pa se pri razvoju aplikacij uporabljajo tudi razna orodja za razvoj programske opreme (SDK - Software development kit), kar zahteva od razvijalca, da se nauči mnogo razvojnih tehnik, preden je sposoben v celoti razviti mobilno aplikacijo. To je predstavljalo oviro večini spletnih razvijalcev, ki so želeli vstopiti v svet razvoja mobilnih aplikacij. Iz tega se je razvila ideja hibridne mobilne aplikacije. Hibridna mobilna aplikacija je izvirna mobilna aplikacija, napisana v spletnih tehnologijah, ki vsebuje

izolirano instanco brskalnika, v kateri teče koda spletne aplikacije.

1.1 Vrste razvoja mobilnih aplikacij

Razvoja mobilnih aplikacij se lahko lotimo na tri različne načine. Ti načini so razvoj izvirnih mobilnih aplikacij, razvoj mobilnih spletnih strani (aplikacij) in razvoj hibridnih mobilnih aplikacij. Vsak ima svoje prednosti in slabosti. Na sliki 1.1 so prikazane razlike v arhitekturi in primer, kako bi posamezna aplikacija dostopala do podatkovne baze oziroma spletnega servisa.

Slika 1.1: Prikaz treh različnih načinov razvoja mobilne aplikacije [3].

1.1.1 Razvoj izvirne mobilne aplikacije

Za razvoj izvirne mobilne aplikacije moramo napisati kodo v privzetem jeziku izbrane platforme. Za Android je to Java, za iOS objektni C, za Windows Phone pa C#. Razvijalec prevede kodo in jo namesti na napravo. Z uporabo orodij SDK posamezne platforme aplikacija neposredno komunicira z aplikacijskim programskim vmesnikom (API - Application programming interface) operacijskega sistema, ki omogoča dostop do podatkovnega diska, komunikacijo preko interneta, uporabo zunanjih sistemov kot je sistem GPS, itd.

Tabeli 1.1 in 1.2 prikazujeta prednosti oziroma slabosti izvirnih mobilnih aplikacij. Glavna prednost izvirnih mobilnih aplikacij je tesna povezanost s platformo, glavna slabost pa je, da tovrsten razvoj zahteva veliko vloženega truda, prav tako pa je zahtevno tudi vzdrževanje. Takšen razvoj je najbolj primeren za razvijalce, ki obvladajo izviren jezik, ali ekipe z obsežnimi viri in potrebo po prednostih izvirne mobilne aplikacije.

Prednost	**Opis**
Namenski vmesnik API	Aplikacija neposredno komunicira z vmesnikom API operacijskega sistema, kar zagotavlja tesno povezavo z operacijskim sistemom.
Hitrost	Zaradi zgoraj omenjene povezave lahko takšne aplikacije dosežejo maksimalno odzivnost.
Isto okolje	Napisane so v istem okolju kot ciljna platforma, kar olajša delo razvijalcem, ki se spoznajo na to okolje.

Tabela 1.1: Prednosti izvirnih mobilnih aplikacij.

Slabost	**Opis**
Jezikovne zahteve	Aplikacija zahteva znanje platformnega jezika in znanje, kako uporabiti izviren vmesnik API.
Ni večplatformna	Mogoče jih je razviti le za eno platformo naenkrat.
Visoka stopnja napora	Ponavadi zahtevajo večji napor za razvoj, kar poveča stroške.

Tabela 1.2: Slabosti izvirnih mobilnih aplikacij.

1.1.2 Razvoj mobilne spletne strani (aplikacije)

Mobilne spletne strani ali mobilne spletne aplikacije so spletne strani, do katerih dostopamo preko brskalnika, ki je nameščen na mobilni napravi. Oblikovane so tako, da se primerno obnašajo ob različnih širinah zaslona (to je pomembno predvsem na majhnih zaslonih mobilnih naprav, kjer uporabnik nima možnosti določanja velikosti okna). Takšni tehniki pravimo odzivni dizajn (*ang. responsive design*). Vsebina spletne strani se povečuje, zmanjšuje in premika po zaslonu glede na velikost zaslona.

Tabeli 1.3 in 1.4 prikazujeta prednosti oziroma slabosti mobilnih spletnih strani. Glavna prednost mobilnih spletnih strani je nizka raven truda in možnost izvajanja na različnih mobilnih napravah, glavna slabost pa življenje znotraj brskalnika, kar prinaša več omejitev in slabosti.

Prednost	**Opis**
Vzdrževanje	Enostavno vzdrževanje in posodabljanje, brez postopkov odobritve in namestitve preko različnih kanalov.
Brez namestitve	Do spletnih strani dostopamo preko interneta. Ne zahtevajo namestitve na mobilno napravo.
So večplatformne	Vsaka mobilna naprava ima brskalnik, kar pomeni, da je spletna stran dosegljiva s katerekoli mobilne naprave.

Tabela 1.3: Prednosti mobilnih spletnih strani.

Slabost	**Opis**
Ni dostopa do operacijskega sistema	Ker tečejo v brskalniku, nimajo dostopa do operacijskega sistema mobilne naprave.
Za zagon je potrebna tipkovnica	Da uporabnik zažene aplikacijo, jo mora najprej poiskati na internetu oziroma vtipkati njen naslov.
Omejen nabor gradnikov	Težko je ustvariti na dotik prijazne gradnike, še posebej, če je stran hkrati oblikovana za namizno uporabo.
Upad uporabe	Količina časa, ki ga uporabniki preživijo na mobilnem spletu, upada, medtem ko se uporaba mobilnih aplikacij povečuje[3].
Različni brskalniki	Vsebina spletne strani se lahko slabše obnaša oziroma prikaže na nekaterih brskalnikih.

Tabela 1.4: Slabosti mobilnih spletnih strani.

1.1.3 Razvoj hibridne mobilne aplikacije

Hibridna mobilna aplikacija je vrsta izvirne mobilne aplikacije, ki vsebuje izolirano instanco brskalnika imenovano *WebView*, v kateri teče koda spletne

aplikacije. Uporablja ovojnico izvirne mobilne aplikacije, ki služi kot komunikacija med operacijskim sistemom in instanco brskalnika. To pomeni, da lahko spletne aplikacije zaganjamo na mobilnih napravah brez uporabe brskalnika, hkrati pa uporabljamo funkcionalnosti naprave kot je sistem GPS ali kamera.

Orodja, ki omogočajo komunikacijo med *WebView* in operacijskim sistemom, so razlog za obstoj hibridnih mobilnih aplikacij. Ta orodja niso del uradnih knjižnic platform Android ali iOS, ampak so neodvisni in ponavadi odprtokodni projekti, kot je Apache Cordova, ki je bil uporabljen tudi pri tej diplomi. Ko se hibridna mobilna aplikacija prevede, se spletna aplikacija pretvori v izvirno mobilno aplikacijo.

Hibridne mobilne aplikacije zagotavljajo trdno podlago za razvoj mobilnih aplikacij, ob tem pa še vedno lahko uporabljamo prednosti spletne platforme. Večino aplikacije lahko napišemo kot spletno aplikacijo. Kadarkoli potrebujemo dostop do funkcionalnosti naprave, nam je to omogočeno kar z JavaScript kodo.

Tabeli 1.5 in 1.6 prikazujeta prednosti oziroma slabosti hibridnih mobilnih aplikacij.

Prednost	**Opis**
So večplatformne	Aplikacijo razvijemo samo enkrat in jo lahko nato prevedemo in namestimo na različne platforme.
Uporaba spletnega znanja	Razvoj je mogoč z znanji, ki jih uporabljamo pri razvoju spletnih aplikacij.
Dostop do funkcionalnosti naprave	Ker *WebView* živi znotraj namenske aplikacije, imamo dostop do različnih funkcionalnosti naprave.
Hiter razvoj	Ni potrebnega konstantnega prevajanja aplikacije med razvojem. Uporabljamo lahko enaka razvojna orodja, kot pri razvoju spletne aplikacije.

Tabela 1.5: Prednosti hibridnih mobilnih aplikacij.

Slabost	**Opis**
WebView omejitve	Odzivnost aplikacije je odvisna od hitrosti *WebView* instance. To pomeni, da je kvaliteta obnašanja aplikacije odvisna od brskalnika, ki je nameščen na mobilni napravi.
Dostop do funkcionalnosti operacijskega sistema preko vtičnikov	Dostop do funkcionalnosti operacijskega sistema je odvisen od razpoložljivosti vtičnikov, napisanih za določeno funkcionalnost naprave.
Ni izvirnih gradnikov	Ne moremo uporabiti izvirnih komponent za uporabniški vmesnik.

Tabela 1.6: Slabosti hibridnih mobilnih aplikacij.

1.2 Ideja za razvoj mobilne aplikacije

Za primer prikaza razvoja hibridne mobilne aplikacije smo se odločili razviti aplikacijo, ki bo predvsem uporabna in ne bo zavržena takoj po končani diplomi. Zahtevali smo tudi, da aplikacija uporablja vsaj eno od funkcionalnosti operacijskega sistema, s čimer bi pokazali eno izmed prednosti hibridne mobilne aplikacije.

Sama ideja se mi je porodila, ko sem v neznanem predelu Ljubljane iskal najbližjega ponudnika študenstke prehrane. Ugotovil sem, da v trgovini Play takrat ni bilo aplikacije, ki bi mi hitro in enostavno poiskala primerne restavracije.

Tako smo se odločili razviti mobilno aplikacijo, ki bo uporabniku hitro in enostavno prikazala najbližje ponudnike študentske prehrane. Pri postavljenih zahtevah pa je nujno potrebno, da razvita aplikacija uporablja funkcionalnost operacijskega sistema (trenutna lokacija). Zahtevali smo tudi, da aplikacija omogoča pregled in iskanje med vsemi ponudniki študetnske pre-

hrane, ki so trenutno na voljo. Za naziv aplikacije sem izbral ime NaBon, saj aplikacija odgovarja na vprašanje: "Kam vse lahko grem jesti na študentski bon?".

1.2.1 Pregled trenutnih podobnih rešitev

V tabelah 1.7, 1.8 in 1.9 so naštete trenutne podobne rešitve, ki obstajajo za platformi iOS (App Store[1]) in Android (Trgovina Play[2]) ter rešitve, ki so dosegljive na spletu preko brskalnika mobilne naprave. Ob vsaki rešitvi je podan komentar na funkcionalnost iskanja najbližjih ponudnikov študentske prehrane in datum zadnje nadgradnje.

Naziv	**Komentar**	**Datum**
ŠUOPit! [20]	Prikazan je seznam najbližjih ponudnikov. Ni zemljevida z vsemi ponudniki. Prikaz lokacije ponudnika in poti do njega ne dela ob vsakem kliku.	9.6.2016
mobileVŠ [21]	Ni seznama najbližjih ponudikov. Na zemljevidu sta prikazana samo dva najbližja ponudnika. Na zemljevidu so podvojeni prikazi lokacij ponudnikov.	2.2.2016
ehrana.si [22]	Ni zemljevida. Nekateri ponudniki manjkajo.	30.11.2015

Tabela 1.7: Pregled rešitev, ki so objavljene v trgovini Play.

Naziv	**Komentar**	**Datum**
Prehrana [23]	Ni zemljevida z vsemi ponudniki.	2.11.2015
ehrana.si [23]	Ni zemljevida. Nekateri ponudniki manjkajo.	15.11.2012

Tabela 1.8: Pregled rešitev, ki so objavljene v trgovini App Store.

[1]https://itunes.apple.com/si/genre/mac/id39?mt=12
[2]https://play.google.com/store

Naziv	**Komentar**	**Datum**
Bonar [24]	/	Ni znan.
Študentska prehrana [25]	Ni zemljevida.	2010

Tabela 1.9: Pregled spletnih rešitev.

Poglavje 2

Opis uporabljenih tehnologij

V tem poglavju so opisane vse tehnologije in orodja, ki so bila uporabljena pri izdelavi te diplome.

2.1 Spletne tehnologije

Najprej so predstavljene glavne tehnologije, s katerimi je bila razvita mobilna hibridna aplikacija. To so HTML, JavaScript in CSS, ki skupaj tvorijo temelj spletnih tehnologij.

2.1.1 HTML

HTML je standardni označevalni jezik za ustvarjanje spletnih strani ter oblikovanje uporabniških vmesnikov za spletne aplikacije. HTML semantično opisuje strukturo spletnih strani z elementi HTML. Ti so naslovi, odstavki, seznami, povezave in še mnogi drugi. V dokument HTML lahko vgradimo skripte, napisane v jeziku JavaScript, ki določajo obnašanje spletne strani in omogočajo dinamičnost strani. S CSS pa natančno določimo videz in postavitev elementov.

2.1.2 CSS

CSS je slogovni jezik, ki se uporablja za določitev opisa predstavitvenih dokumentov, napisanih v označevalnem jeziku. Večinoma se uporablja za nastavitev vizualnega sloga spletnih strani in uporabniških vmesnikov napisanih v jeziku HTML, lahko ga pa uporabimo tudi za katerikoli dokument XML, npr. SVG.

CSS je namenjen predvsem ločitvi vsebine dokumenta in njegove predstavitve. Ta ločitev povečuje nadzor in berljivost predstavitve ter omogoča, da si lahko več dokumentov HTML deli isti slogovni opis.

2.1.3 JavaScript

JavaScript je dinamično tipiziran skriptni programski jezik. Je standardiziran v specifikaciji ECMAScript [7]. JavaScript ima vmesnik API za delo s tekstom, polji, datumi in regularnimi izrazi, ne vsebuje pa vhodno/izhodnih (I/O - input/output) funkcij, kot so funkcije za delo z omrežjem ali dostop do podatkovnega diska. Za njih poskrbi okolje gostitelja, v katerem se izvaja.

JavaScript se tudi uporablja v okoljih, ki ne temeljijo na spletu. To so naprimer dokumenti PDF, namizni pripomočki, pri razvoju iger ter pri razvoju mobilnih in namiznih aplikacij [5]. V porastu je uporaba v zalednih strežniških aplikacijah, ki delujejo v okoljih, kot je Node.js. V okolje Node.js je samodejno vključen tudi upravitelj paketov *npm*, ki omogoča njihovo distribucijo. Ob pisanju te diplome je bilo v *npm* registru na voljo kar 320.000 paketov [18]. Med temi paketi najdemo tudi pakete LoopBack, Ionic in Apache Cordova.

2.2 Ionic

Ionic je kombinacija tehnologij in pripomočkov, s katerimi lahko hitro in enostavno razvijemo hibridne mobilne aplikacije. Ionic je nadgradnja tehnologij AngularJS in Apache Cordova. S prvim razvijemo spletno aplikacijo, z dru-

gim pa spletno aplikacijo pretvorimo v izvirno mobilno aplikacijo. Ponuja nam komponente uporabniškega vmesnika, ki jih pogosto najdemo pri mobilnih aplikacijah, vendar manjkajo v zbirki standardnih elementov HTML.

Je odprtokoden projekt, ki ga je razvilo podjetje Drifty [9]. Prva verzija je izšla novembra 2013. Njegova priljubljenost narašča, saj je s pomočjo tega ogrodja razvitih kar 20.000 aplikacij vsak mesec [9]. V beta razvoju je tudi že Ionic2, ki temelji na tehnologiji Angular2 ter standardu spletnih komponent (*ang. web components*).

Ionic ima tudi orodje, ki je dostopno preko ukazne vrstice (CLI - Command-line interface) in služi kot razvojno okolje, s katerim lahko hitro in enostavno generiramo začetne projekte, zaženemo in testiramo aplikacije itd.

2.2.1 AngularJS

AngularJS (znan tudi kot samo Angular) je JavaScript orodje za ustvarjanje odjemalskih in zalednih spletnih aplikacij. Razvilo ga je podjetje Google z namenom, da bi olajšal razvoj eno-stranskih (*ang. single-page*) spletnih aplikacij. Zagotavlja model-pogled-kontroler (MVC - Model-View-Controller) arhitekturo aplikacij in elemente, ki jih uporabljamo v poslovnih spletnih aplikacijah.

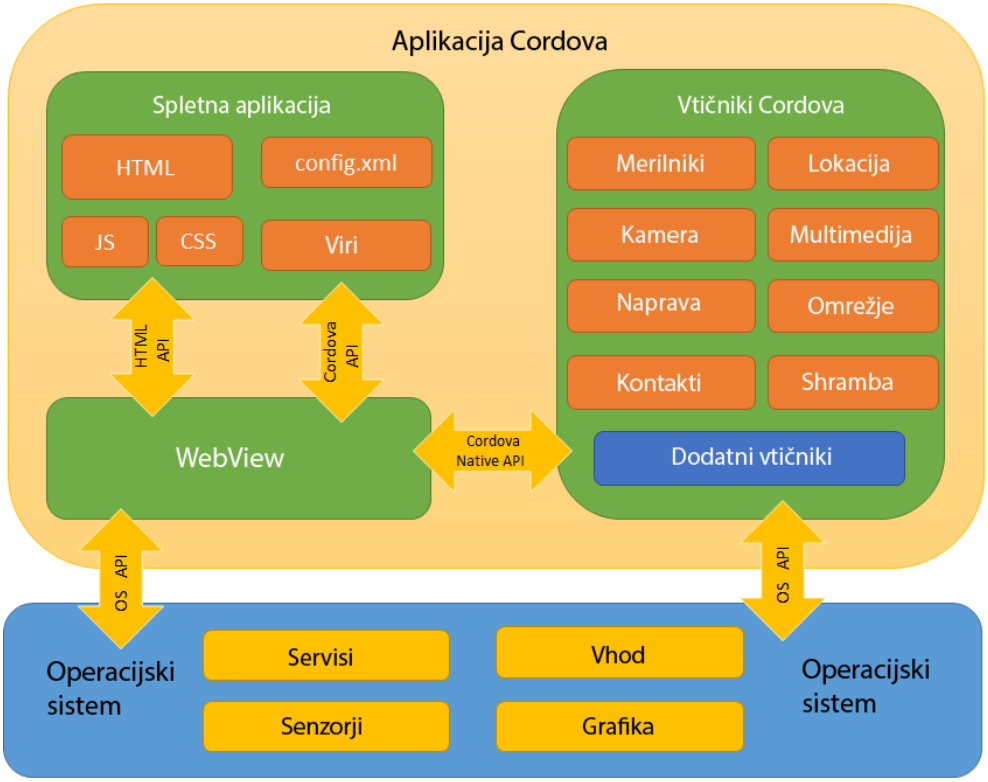
Angular uporablja vzorec imenovan injiciranje odvisnosti (*ang. dependency injection*), s katerim v posamezne module vključujemo servise, ki so definirani v drugih modulih. To zagotavlja, da so Angular aplikacije sestavljene iz večih smiselno zaključenih celot (modulov).

2.2.2 Apache Cordova

Cordova je orodje za izdelavo hibridne mobilne aplikacije. Skrbi za komunikacijo med oknom brskalnika in vmesnikom API operacijskega sistema. Zagotavlja sistem vtičnikov, ki omogočajo odjemalcu dostop do funkcionalnosti operacijskega sistema kot je kamera, luč LED, sistem GPS itd. Eden izmed teh vtičnikov je Cordova GoogleMaps, ki nam omogoča dostop do Google

Maps SDK v2 na Android napravah ter Google Maps SDK na iOS napravah.

Na sliki 2.1 je prikazana arhitektura hibridne mobilne aplikacije, ki je izdelana s pomočjo orodja Cordova.



Slika 2.1: Arhitektura Cordova aplikacije [26].

2.2.3 Ionic.io

Ionic.io je oblačna storitev, ki razširja celotno Ionic ogrodje. Ponuja nam vrsto zalednih storitev ter orodij:

- Push - pošiljanje *push* obvestil;
- Deploy - posodabljanje aplikacij na zahtevo;
- Analytics - analiza uporabe aplikacije;
- Package - gradnja paketov za platformo Android ali iOS.

2.3 LoopBack

LoopBack orodje sestavlja množica Node.js modulov, ki jih lahko uporabimo skupaj ali samostojno, za hitro gradnjo aplikacij, ki izpostvaljajo predstavitvene prenose stanja (REST - Representational state transfer) vmesnikov API. Je odprtokoden in razširljiv. Njegove glavne funkcionalnosti so:

- ustvarjanje dinamičnih REST vmesnikov API za povezovanje končnih točk (*ang. end-to-end*) z malo ali brez kodiranja;
- dostop do podatkov relacijskih podatkovnih baz (SQL Server, MySQL, Oracle, PostgreSQL), podatkov ne-relacijskih baz (MongoDB), podatkov preko REST vmesnikov API in mnogo drugih virov;
- vključitev modela odvisnosti med podatki in kontrole dostopa za kompleksne vmesnike API;
- dodatne komponente za shranjevanje datotek, zunanjo prijavo in OAuth 2.0 podporo;
- enostavno ustvarjanje odjemalskih aplikacij, z uporabo Android SDK, iOS SDK ali JavaScript SDK;
- zagon aplikacije v oblaku ali lokalno.

2.4 MongoDB

MongoDB je večplatformna NoSQL, se pravi ne-relacijska, podatkovna baza. Je dokumentno orientirana in uporablja strukturo BSON. Struktura BSON je binarni format zapisa dokumentov v strukturi objektne notacije JavaScript (JSON - JavaScript Object Notation) z dinamično shemo.

2.5 Heroku

Heroku je oblačna storitev tipa platforma kot storitev (PaaS), ki podpira programske jezike kot so Node.js, Ruby, Java, PHP, Python itd. Hkrati z gostovanjem aplikacije nam Heroku ponuja zbirko dodatkov, ki jih lahko namestimo na platformo. Eden izmed njih je mLab. mLab je grafični vmesnik za upravljanje z bazo MongoDB, ki jo zagotavlja Heroku.

Poglavje 3

Razvoj aplikacije in njene funkcionalnosti

V tem poglavju je opisan razvoj hibridne mobilne aplikacije. Predstavljena je podatkovna baza, ki hrani potrebne podatke, srednji sloj, ki zagotavlja zaledje aplikacije, funkcionalnosti aplikacije in razvoj posameznih delov, iz katerih je sestavljena aplikacija.

3.1 Podatkovna baza

Na začetku razvoja aplikacije je bilo mišljeno, da se bo podatkovna baza ponudnikov študentske prehrane zgradila ročno, s prepisovanjem podatkov z uradne spletne strani študentske prehrane. Vendar pa smo med razvojem naleteli na projekt Bonar [16], ki je kot nalašč za potrebe aplikacije. Bonar je odprtokoden projekt, ki spada pod slovensko združenje projektov imenovano opendata.si [17]. Ta temelji na ideji odprtih podatkov, ki pravi, da smo za določen sklop podatkov lastniki vsi prebivalci Slovenije in so posledično podatki brez licence.

Bonar zagotavlja vmesnik API, preko katerega lahko dostopamo do podatkov, ki so na uradni spletni strani študentske prehrane. Bonar vsako jutro ob 5:00 po Greenwiškem srednjem času prebere seznam vseh ponudni-

kov študentske prehrane in njihove podrobnosti ter jih zapiše v svojo bazo. Postopek traja približno 10 minut. Vmesnik vrača podatke v obliki JSON. Podrobnosti posameznega ponudnika vsebujejo: ime restavracije, naslov restavracije, telefon, ceno obroka, koordinate restavracije, odpiralni čas, meni in seznam dodatnih storitev oziroma ponudb. Med dodatne storitve sodijo možnost dostave hrane, solatni bar, dostop za invalide, dostop za invalide do toalete, vegetarijanska prehrana, ponudba pic, ponudba kosil ter ponudba hitre hrane.

Aplikacija, ki smo jo razvili, ne uporablja neposredno tega vmesnika. Ta vmesnik se uporablja samo za začetno polnjene naše podatkovne baze na strežniku in vsakodnevno osveževanje podatkov. Za potrebe aplikacije je bilo posebej postavljeno zaledje ter podatkovna baza. Na ta način se zmanjša odvisnost projekta od drugih. Za ponudnika podatkovne baze je bil izbran MongoDB, ki ga gostuje oblačna storitev Heroku.

Podatkovna baza vsebuje eno zbirko (*ang. collection*) dokumentov (*ang. document*) imenovano Restaurants. Dokumenti v zbirki so zapisani v strukturi JSON. Posamezen dokument je sestavljen iz parov ključ-vrednost (*ang. key-value*). Iz slike 3.1 je razvidna struktura posameznega dokumenta, nazivi ključev in podatkovni tipi vrednosti.

3.2 Srednji sloj

Zaledje aplikacije je zgrajeno v okolju Node.js. Vmesnik API aplikacije je bil zgrajen z uporabo knjižnice LoopBack. Najprej je bil kreiran model Restaurant, v katerem so definirane vse lastnosti. Vsaka lastnost ima svoj tip. LoopBack je nato sam ustvaril REST vmesnik API, ki ponuja vse standardne metode za ustvarjanje, branje, posodabljanje in brisanje (CRUD - create, read, update, delete) podatkov v podatkovni bazi. Za potrebe odjemalca je potrebna samo ena metoda GET, ki nam vrne seznam vseh restvracij z njihovimi podrobnostmi, zato so bile ostale metode onemogočene. Strežnik, ki servira vmesnik API, teče v oblačni storitvi Heroku.

```
{
    "_id": Number,
    "name": String,
    "address": String,
    "telephone": Array of Strings,
    "price": String,
    "opening": {
        "week": Boolean or Array of Strings,
        "saturday":  Boolean or Array of Strings,
        "sunday":  Boolean or Array of Strings
    },
    "menu": Array of Arrays of Strings,
    "coordinates": Array of Strings,
    "features": [
        {
            "id": Number,
            "title": String
        },
        ...
    ]
}
```

Slika 3.1: Struktura posameznega dokumenta.

Spodaj je predstavljena koda modula, ki skrbi za vsakodnevno osveževanje podatkov.

```
var CronJob = require('cron').CronJob;
require('log-timestamp');
var request = require('request');
module.exports = function(app) {
  new CronJob('30 5 * * * *', function() {
    console.log('GETTING RESTAURANTS...');
    request('http://bonar.si/api/restaurants', function(error
       , response, body) {
      if (!error && response.statusCode == 200) {
        var restaurants = JSON.parse(body);
        console.log('SUCCESS GETTING RESTAURANTS: ' + 'n=' +
           restaurants.length);
        if (restaurants.length > 0) {
          var model = app.models.Restaurant;
          console.log('CLEARING DATABASE...');
          model.destroyAll({}, function(err, info) {
            if (err) {
              console.log('ERROR CLEARING DATABASE: ' + err);
            } else {
              console.log('SUCCESS CLEARING DATABASE: n=' +
                 info.count);
              console.log('IMPORTING RESTAURANTS...');
              model.create(restaurants, function(err, models)
                  {
                if (err) {
                  console.log('ERROR IMPORTING RESTAURANTS: '
                      + err);
                } else {
                  console.log('SUCCESS IMPORTING RESTAURANTS:
                      n=' + models.length);
                }
              })
            }
          })
        }
```

```
30        } else {
31          console.log('ERROR GETTING RESTAURANTS' + response.
                statusCode + ': ' + error);
32        }
33      });
34    }, null, true, 'GMT');
35  };
```

Na strežniku teče zagonska skripta, ki vsak dan ob 5:30 po Greenwiškem srednjem času, z uporabo *npm* paketov *request* in *cron*, najprej pokliče metodo GET projekta Bonar za pridobitev vseh restavracij (vrstice 5-7). Nato se, če je bil klic uspešen in odgovor vsebuje vsaj en element (vrstice 8-13), pobriše podatkovna baza (vrstica 14). Če je bilo brisanje podatkov uspešno, se baza napolni z novim seznamom ponudnikov (vrstica 20). Tako podatki ostanejo vedno aktualni. Paket *request* (vrstica 3) omogoča klic metode HTTP, paket *cron* (vrstica 1) pa zagotavlja vsakodnevno izvajanje celotnega procesa ob določenem času. Uporabljen je tudi paket *log-timestamp* (vrstica 2), ki skrbi za logiranje s časovnim odtisom ob izvajanju posameznih metod.

3.3 Odjemalec

Mobilna aplikacija je bila razvita v orodju Ionic, ki je sestavljen iz knjižnice AngularJS za razvoj spletne aplikacije ter orodja Cordova za gradnjo hibridnih mobilnih aplikacij. Aplikacija je sestavljena iz treh delov: zemljevida restavracij, seznama restavracij in podrobnosti posamezne restavracije. Zemljevid in seznam se obnašata kot zavihka, do katerih lahko uporabnik dostopa kadarkoli. Podrobnosti posamezne restavracije pa so podstran strani s seznamom restavracij.

Spodaj je predstavljen izsek kode s konfiguracijo Angular aplikacije.

```
1  angular.module('naBon', ['ionic', 'ionic.service.core', '
       ionic.service.analytics', 'ngCordova'])
2    .config(function($stateProvider, $urlRouterProvider) {
3      $stateProvider
4        .state('tabs', {
```

```
5         url: "/tab",
6         abstract: true,
7         templateUrl: "templates/tabs.html"
8       })
9       .state('tabs.restaurants', {
10        url: "/restaurants",
11        views: {
12          'restaurants-tab': {
13            templateUrl: "templates/restaurants.html",
14            controller: 'RestaurantsCtrl'
15          }
16        }
17      })
18      .state('tabs.details', {
19        url: "/details",
20        views: {
21          'restaurants-tab': {
22            templateUrl: "templates/details.html",
23            controller: 'DetailsCtrl'
24          }
25        }
26      })
27      .state('tabs.map', {
28        url: "/map",
29        views: {
30          'map-tab': {
31            templateUrl: "templates/map.html",
32            controller: 'MapCtrl'
33          }
34        }
35      });
36    $urlRouterProvider.otherwise("/tab/map");
37  });
```

Iz zgornjega izseka kode (vrstica 1) je razvidno, da so v aplikacijo injicirani štirje moduli:

- ***ionic***, ki zagotavlja potrebno osnovno Angular kodo, dodatne *de-facto*

module, kot je *ui-router* za navigacjo po aplikaciji, HTML elemente, primerne za mobilno uporabo ter servise, ki skrbijo za dinamičnost teh elementov;

- ***ionic.service.core***, ki zagotavlja jedro Ionic.io storitev za povezovanje aplikacije z oblačnimi storitvami;

- ***ionic.service.analytics***, ki zagotavlja potrebno logiko za analizo uporabe in atribute, ki jih pripnemo na HTML elemente, pri katerih želimo beleženje uporabe;

- ***ngCordova***, ki zagotavlja Angular ovojnice (*ang. wrappers*) za delo s Cordova vtičniki.

Z uporabo modula *ui-router* in njegovega servisa *$stateProvider* je definirana struktura aplikacije. Iz zgornjega izseka kode je razvidno, da je aplikacija sestavljena iz štirih stanj (vrstice 4, 9, 18 in 27). Vsako stanje (*state*) ima definiran po en pogled (*ang. view*), ki je sestavljen iz predloge HTML (*ang. template*), kjer je definiran izgled pogleda, in JavaScript kontroler (*ang. controller*), kjer je definirana logika obnašanja predloge. Ta stanja so:

- ***tabs.map*** je stanje, ki prikazuje zemljevid restavracij (vrstice 27-35);

- ***tabs.restaurants*** je stanje, ki prikazuje seznam vseh restavracij in iskalnik po tem seznamu (vrstice 9-17);

- ***tabs.details*** je stanje, ki prikazuje podrobnosti posamezne restavracije (vrstice 18-26);

- ***tabs*** je abstraktno stanje, ki se ne prikazuje na zaslonu, ampak skrbi za preklop med stanji *tabs.restaurants* in *tabs.map* (vrstice 4-8). Ker je stanje abstraktno, nima pogleda. Za njegovo logiko delovanja skrbi *ionic* servis *$ionicTabsDelegate*.

V vrstici 36 je tudi razvidno, da je z uporabo servisa *$urlRouterProvider* definirano začetno stanje aplikacije. To je stanje, ki prikazuje zemljevid restavracij.

Ob omenjeni strukturi aplikacije je definiran tudi globalni kontroler *AppCtrl*, kjer so definirane funkcije, ki vsebujejo logiko za povezovanje posameznih delov aplikacij in funkcije, ki jih uporablja več kontrolerjev naenkrat. Definiran je tudi servis *AppService*, ki skrbi za klic metode HTTP za pridobitev seznama restavracij in shranjuje podatek o iskalnem nizu pri seznamu restavracij.

3.3.1 Zemljevid restavracij

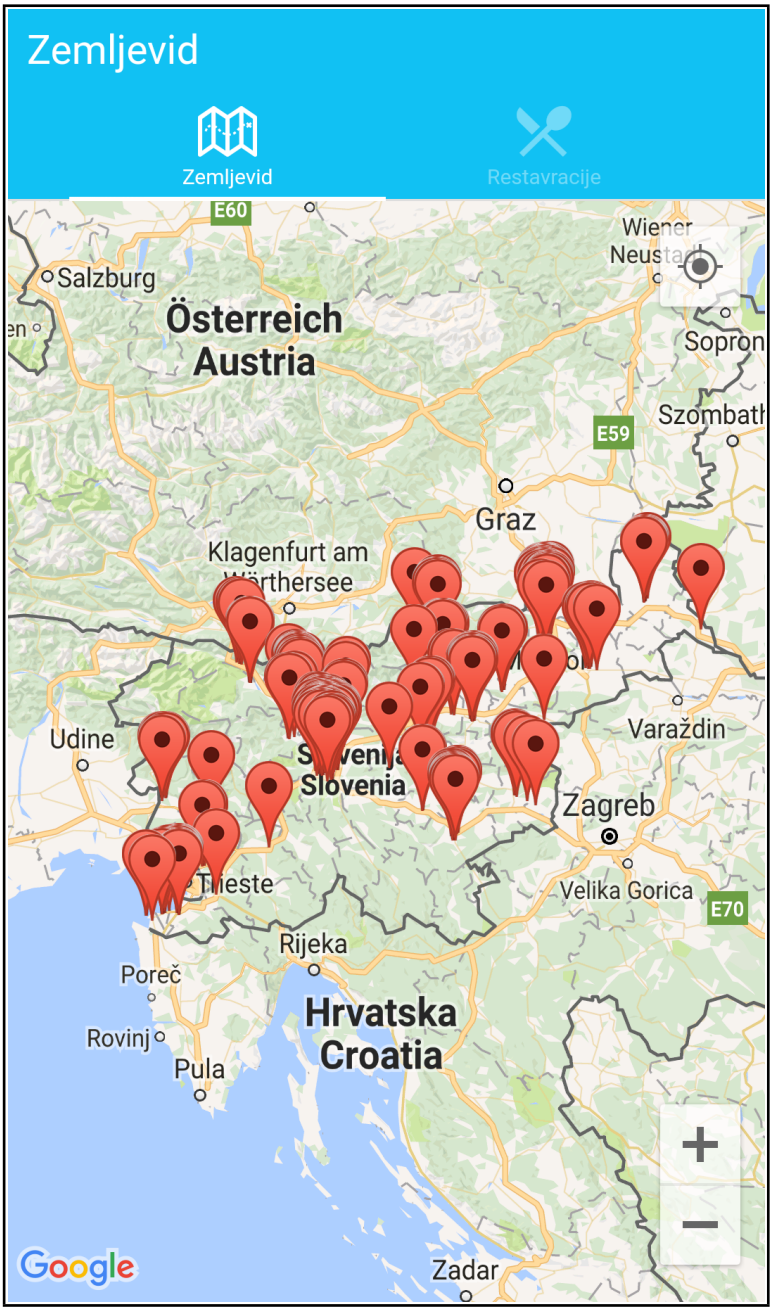
Sedaj bomo opisali stanje za prikaz zemljevida restavracij.

Opis obnašanja funkcionalnosti

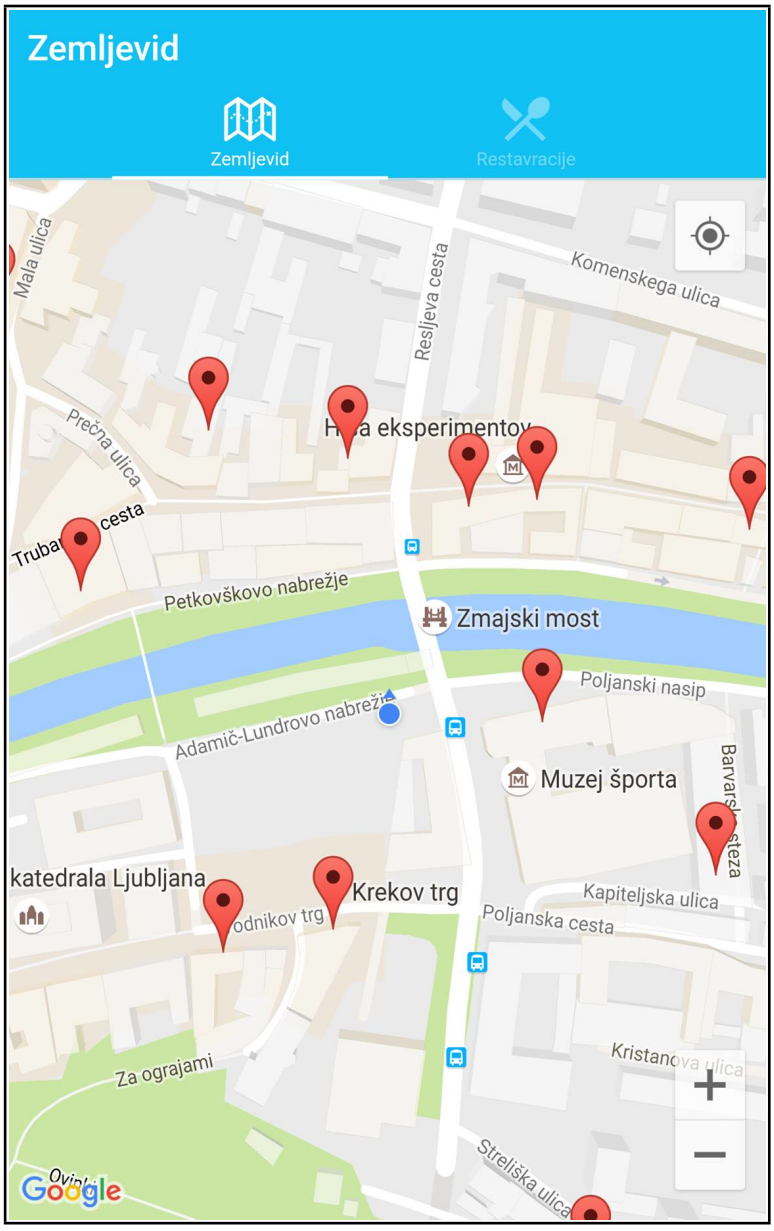
Ob zagonu aplikacije je privzeto aktiven zavihek Zemljevid za prikaz zemljevida restavracij. Če je sistem GPS na napravi trenutno nedosegljiv, vidi uporabnik ob zagonu zemljevid celotne Slovenije, kot je prikazano na sliki 3.2. V nasprotnem primeru vidi uporabnik okolico svoje lokacije s 17 kratno povečavo, kot je prikazano na sliki 3.3.

Uporabniku so na voljo funkcionalnosti zemljevida Google. To so premikanje po zemljevidu, približevanje in oddaljevanje, gumb za postavitev pogleda na trenutno lokacijo, obračanje okoli osi ter poravnavanje smeri neba.

Če uporabnik klikne na eno izmed bucik, se odpre majhno okence, v katerem so napisani naslov in nazivi vseh restavracij, ki se nahajajo na tem naslovu. Hkrati se prikažeta še dve funkciji zemljevida Google, ki zaženeta aplikacijo Google Maps. Prva funkcija sproži prikaz lokacije ter podrobnosti, ki so na voljo, druga pa sproži navigacijo od trenutne do zahtevane lokacije. Ob kliku na okence z nazivom restavracije, aplikacija odpre stran s podrobnostmi te restavracije.



Slika 3.2: Začetni zaslon ob nedosegljivem sistemu GPS.



Slika 3.3: Začetni zaslon ob dosegljivem sistemu GPS.

Opis razvoja funkcionalnosti

Za komponento zemljevida je bil uporabljen Cordova vtičnik *cordova-plugin-googlemaps*, ki ponuja Google Maps Android SDK 2.0 API in Google Maps iOS SDK vmesnik API za prikaz zemljevida in uporabo vseh njegovih funkcionalnosti. Za njegovo uporabo si je potrebno ustvariti in aktivirati ključe API na spletni strani Google, s katerih lahko nato dostopamo do zemljevida Google Developer. Za komunikacijo s sistemom naprave GPS je bil uporabljen *ngCordova wrapper $cordovaGeolocation*, ki ovija Cordova vtičnik *cordova-plugin-geolocation*.

Aplikacija najprej počaka pet sekund, da pridobi GPS lokacijo naprave, nato se inicializira zemljevid ter pokliče metoda za pridobitev seznama vseh restavracij. Podatki se najprej obdelajo: dodajo se lastnosti za kasnejši lažji prikaz podrobnosti o restvraciji, nato se seznam pogrupira po koordinatah posameznega elementa, saj je lahko več ponudnikov v isti stavbi. Na koncu se seznam uredi po nazivu. Ko so vsi ti postopki končani, se na zemljevid dodajo bucike, kjer vsaka predstavlja eno ali več restavracij. Spodaj je prikazana koda funkcije, ki inicializira zemljevid (vrstica 19), pridobi podatke o restavracijah (vrstica 21), jih obdela (vrstica 22) ter na zemljevidu prikaže vse lokacije restavracij (vrstica 23).

```
1  function setMap(location) {
2    var options = {
3      'controls': {
4        'compass': true,
5        'myLocationButton': true,
6        'zoom': true
7      },
8      'camera': {
9        'latLng': location,
10       'zoom': 17
11     }
12   }
13   if (location == null) {
14     options.camera = {
```

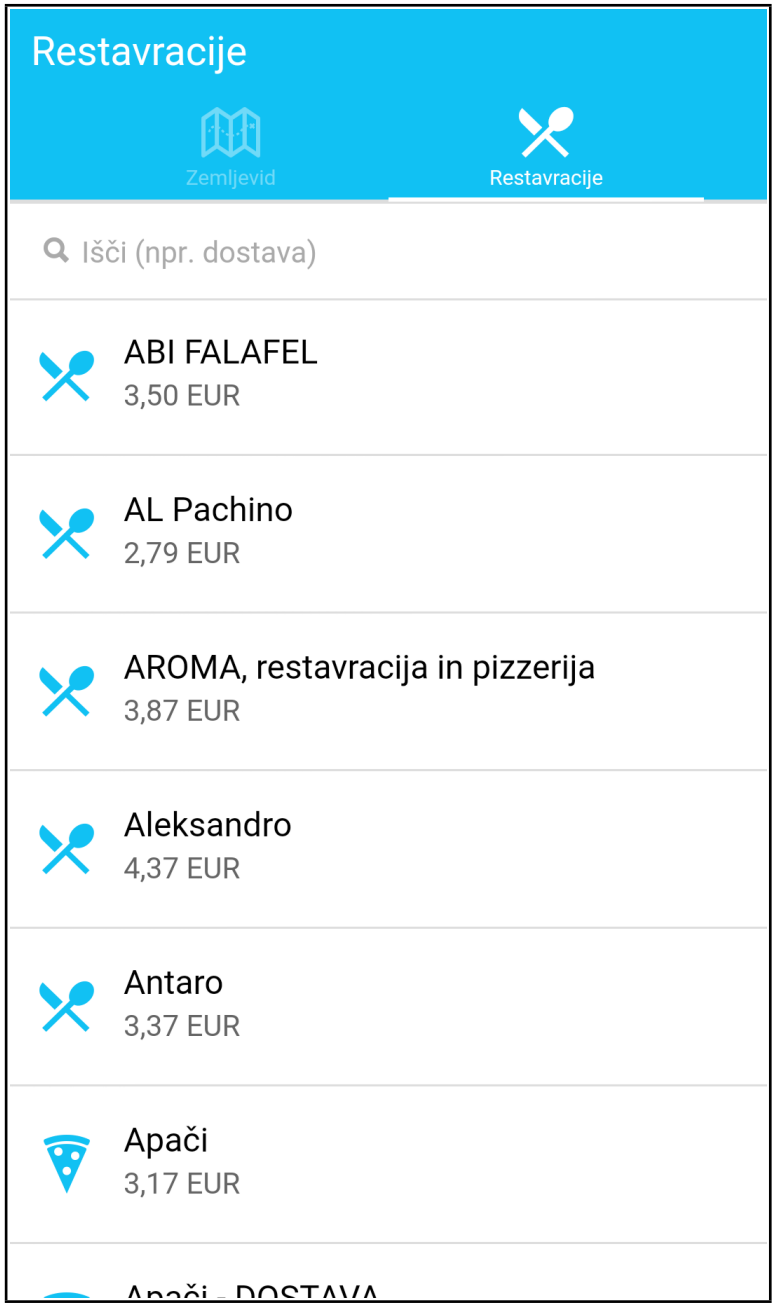
```
15        latLng: new plugin.google.maps.LatLng(46.119944,
              14.815333),
16        zoom: 7
17      }
18    }
19    $rootScope.map = plugin.google.maps.Map.getMap(document.
          getElementById("map_canvas"), options);
20    $rootScope.map.addEventListener(plugin.google.maps.event.
          MAP_READY, function() {
21      AppService.getRestaurants().then(function(success) {
22        $rootScope.restaurants = setData(success.data);
23        addMarkers(groupByLocation($rootScope.restaurants),
              function(markers) {
24          addPositions(markers);
25        });
26      }, function(error) {
27        //empty
28      });
29    });
30  }
```

3.3.2 Seznam restavracij

Sedaj bomo predstavili stanje za izpis seznama restavracij.

Opis obnašanja funkcionalnosti

V zavihku Restavracije sta prikazana iskalnik po vseh restavracijah in rezultat trenutnega iskanja. Ob zagonu aplikacije je iskalni niz prazen. Iskalnik išče po nazivih restavracij. Vsak element v seznamu je predstavljen z ikono, nazivom in ceno obroka, kot je prikazano na sliki 3.4. Če ima restavracija ponudbo pic, je za ikono uporabljena podoba pice, drugače ima element privzeto ikono. Ob kliku na element aplikacija odpre stran s podrobnostmi izbrane restavracije.



Slika 3.4: Seznam restavracij z iskalnikom.

Opis razvoja funkcionalnosti

Za komponento seznama restavracij je uporabljen Ionic element *ion-list*, za katerga skrbi servis *$ionicListDelegate*. V kontroler je injiciran servis *$ionicAnalytics*, ki ob vsakem kliku na posamezen element beleži analizo uporabe. Ob kliku se na oblačno storitev Ionic Analytics pošlje podatek o izbrani restavraciji. Enak podatek se pošlje tudi, kadar uporabnik na zemljevidu klikne na okence posamezne restavracije in se odpre stran s podrobnostmi izbrane restavracije.

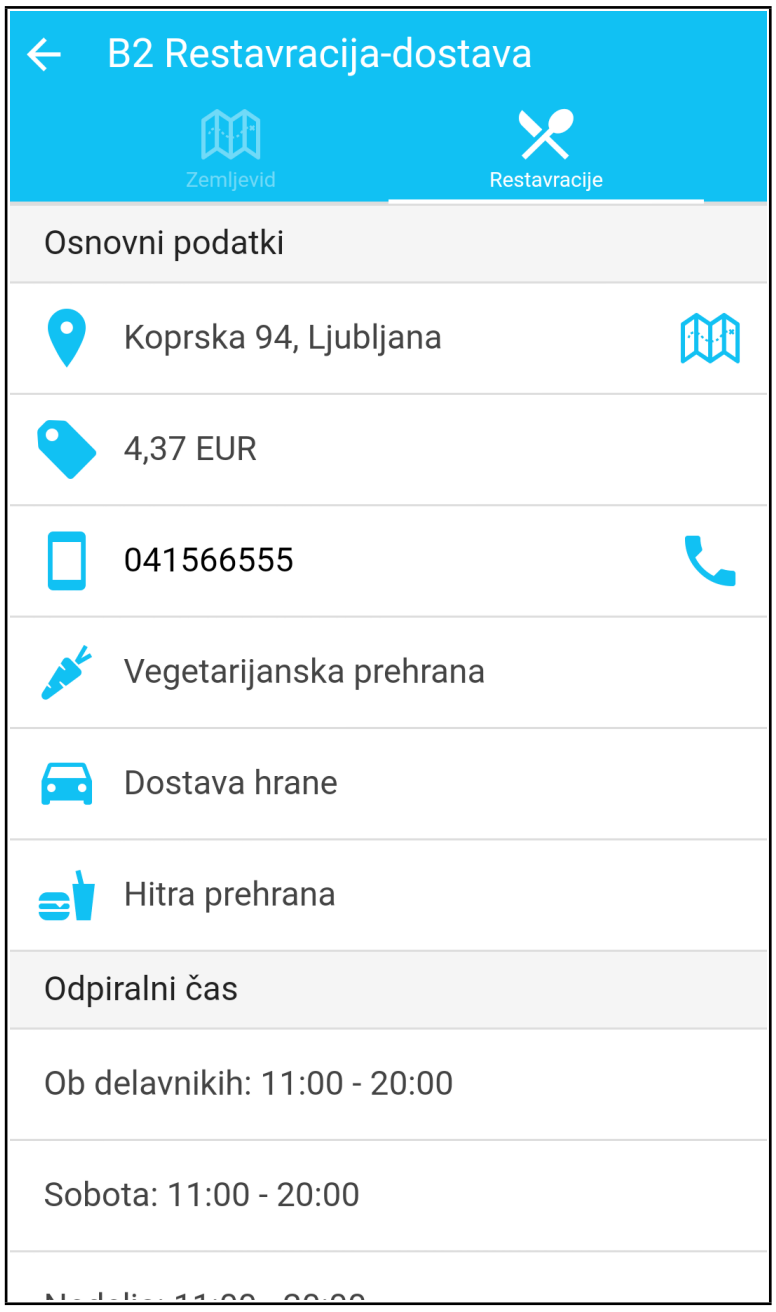
3.3.3 Podrobnosti restavracije

Sedaj bomo opisali še tretje stanje, kjer se prikazujejo podrobnosti izbrane restavracije.

Opis obnašanja funkcionalnosti

Na strani so trije sklopi podatkov: osnovni podatki in odpiralni čas, ki sta prikazana na sliki 3.5, ter jedilnik. V sklopu Osnovni podatki je zapisan naslov restavracije, cena obroka, telefon ponudnika, naštete pa so tudi vse dodatne ponudbe kot so: dostave hrane, solatni bar, dostop za invalide, dostop za invalide do toalete, vegetarjanska prehrana, ponudba pic, ponudba kosil ter ponudba hitre hrane. V sklopu Odpiralni čas so zapisane delovne ure med delovniki, ob sobotah in ob nedeljah. Če ima restvracija kakšno opombo, se le ta tukaj tudi izpiše. V sklopu Jedilnik so različni meniji, ki jih restavracija ponuja.

Elementa z naslovom restavracije in telefonom se obnašata kot gumba. Če uporabnik klikne na element z naslovom restavracije, ga aplikacija preusmeri na zavihek za prikaz zemljevida restavracij in premakne zemljevid s simuliranim premikanjem kamere na zahtevano lokacijo s 17 kratno povečavo. Po končani animaciji se odpre okence z nazivom restavracije ter naslovom. Če uporabnik klikne na element s telefonom ponudnika, se odpre aplikacija naprave za klicanje. V vnosno polje se vpiše telefonska številka ponudnika.



Slika 3.5: Prikaz podrobnosti restavracije.

Opis razvoja funkcionalnosti

Za prikaz podrobnosti restavracije je uporabljen Ionic element *ion-list*, za naslov posameznega sklopa pa Ionic element *ion-header*. Na sliki 3.6 je prikazan element HTML, ki skrbi za prikaz telefonskih številk posamezne restavracije in odpiranje aplikacije za klicanje, ki je nameščena na mobilni napravi. Pri slednjem se uporablja funkcionalnost samega elementa HTML *a*, ki na mobilnih napravah ob dodatni predponi *tel:* poskrbi za odpiranje aplikacije za klicanje.

```
<a ng-href="tel:{{phone}}"
   ng-repeat="phone in selected.telephone"
   class="item item-icon-left item-icon-right">
    <i class="icon ion-android-phone-portrait calm"></i>
        {{phone}}
    <i class="icon ion-android-call calm"></i>
</a>
```

Slika 3.6: Element HTML za prikaz telefonske številke restavracije.

3.3.4 Ikona aplikacije

Ikona aplikacije, ki je prikazana na sliki 3.7, je bila narisana v orodju za delo z vektorsko grafiko. Za primarno barvo ikone je bila izbrana barva, ki dominira v aplikaciji (#11C1F3).

Slika 3.7: Ikona aplikacije.

Poglavje 4

Objava aplikacije in analiza uporabe

V tem poglavju je opisan postopek objave aplikacije na distribucijske kanale posameznih platform. Predstavljena je tudi analiza uporabe aplikacije.

4.1 Objava aplikacije

Po končanem razvoju aplikacije so bile definirane vse nastavitve aplikacije, ki so potrebne za gradnjo distribucijskih paketov. Nastavitve se definirajo v *config.xml* datoteki, katero Cordova nato uporabi pri gradnji paketov za platformi iOS in Android. Glavne nastavitve so *bundle id*, ki določa enoličen identifikator mobilne aplikacije, *name*, ki je naziv aplikacije ter *version*, ki definira verzijo distribucije.

4.1.1 Trgovina Play (Android)

Pred objavo aplikacije v trgovino Play je bilo potrebno najprej generirati produkcijsko gradnjo aplikacije. Za to je poskrbela Cordova, ki spletno aplikacijo pretvori v hibridno mobilno aplikacijo. Rezultat je nepodpisana *.apk* datoteka.

Nepodpisano produkcijsko gradnjo je nato potrebno podpisati s privatnim ključem. Za generiranje ključev in podpisovanje se uporabljajo orodja za razvoj programske opreme Android SDK. Ko imamo pripravljeno podpisano produkcijsko gradnjo, jo lahko objavimo preko spletne aplikacije Google Play Store Developer Console, kamor se prijavimo s svojim Google Developer računom.

4.1.2 Trgovina App Store (iOS)

Produkcijsko gradnjo aplikacije za platformo iOS lahko generiramo samo na platformi OS X. Za gradnjo hibridne mobilne aplikacije poskrbi Cordova. Za nadaljno delo potrebujemo orodje Xcode, v katerega se vpišemo s svojim Apple Developer računom in zapakiramo aplikacijo v arhiv, ki je pripravljen za objavo. Pri pakiranju je potrebno v projekt dodati produkcijski certifikat, s katerim podpišemo aplikacijo.

Rezultat je *.ipa* datoteka, ki jo lahko objavimo preko spletne aplikacije iTunnes Connect.

4.2 Analiza uporabe

Za analizo uporabe aplikacije je bila uporabljena oblačna storitev Ionic Analytics. V aplikacijo je bilo vgrajeno sledenje dvema dogodkoma:

- Uporabnik zažene aplikacijo.
- Uporabnik odpre podstran s podrobnostmi posamezne restavracije.

Ko uporabnik odpre podstran s podrobnostmi o restavraciji, se hkrati pošljeta tudi naziv restavracije in njen enolični identifikator. Tako lahko iz pridobljenih podatkov ustvarimo statistiko o najbolj priljubljenih restvracijah. V nadaljevanju sta predstavljeni dve analizi uporabe: pregled najbolj priljubljenih restavracij in frekvenca uporabe aplikacije.

4.2.1 Pregled najbolj priljubljenih restavracij

V tabeli 4.1 je zapisanih 10 najbolj priljubljenih izmed vseh restavracij, ki ponujajo študentsko prehrano. Vseh restavracij je v času pisanja diplomske naloge bilo 504. V desnem stolpcu je zapisan delež klikov na posamezno restavracijo. Analiza zajema uporabo aplikacije od začetne objave aplikacije 1.7.2016 do 20.8.2016. Število vseh klikov v tem obdobju je bilo 392. Vidimo lahko, da imamo eno restavracijo, ki močno prednjači pred vsemi ostalimi. Na podlagi trenutnega števila vseh klikov na posamezne restavracije pa je težko izvleči kakšne relevantne zaključke.

Naziv	**Delež**
Kitajska restavracija Sonce	5,0%
Pek Matjaž Trubarjeva	3,3%
City grill	3.0%
Kitajska restavracija Zvezda	2,6%
Cantante cafe center	2,3%
Gostilna pod Škalcami - DOSTAVA	2,0%
Pizzeria FoculuS	2,0%
NANA kavarna & lounge bar	2,0%
Aleksandro	1,7%
ŽITO Ljubljana Vodnik	1,7%

Tabela 4.1: Pregled najbolj priljubljenih restavracij.

4.2.2 Frekvenca uporabe aplikacije

Na sliki 4.1 je prikazan graf frekvence uporabe aplikacije. Analiza zajema uporabo aplikacije med 1.7.2016 in 20.8.2016. Posamezna točka na grafu predstavlja število zagonov aplikacije na določen dan. Frekvenca uporabe se je zmanjšala po 15.7.2016, čemur je zagotovo pripomoglo dejstvo, da se je takrat zaključilo letno izpitno obdobje. V avgustu pa se je ponovno povečala, kar sovpada s časom prvih vpisov v študijske programe.

Zaradi časa objave aplikacije, ki je prišel tik pred začetkom poletnih počitnic, je prikaz uporabe aplikacije pomanjkljiv in ga ne moremo posplošiti na obdobje ko se izvajajo predavanja.

Slika 4.1: Frekvenca uporabe aplikacije za obdobje od 1.7.2016 do 20.8.2016.

Poglavje 5

Sklepne ugotovitve

Razvili smo aplikacijo, ki služi kot pomoč pri enostavnem iskanju ponudnikov subvencionirane prehrane za študente.

5.1 Zaključek

Rezultat diplomskega dela je mobilna aplikacija, ki je dosegljiva na obeh glavnih mobilnih platformah (iOS in Android). Je hibridna mobilna aplikacija, kar pomeni, da je bila napisana v spletnih tehnologijah, vendar se obnaša kot izvirna mobilna aplikacija in hkrati uporablja funkcinalnosti operacijskega sistema (sistem GPS). Aplikacija je bila razvita s pomočjo orodja Ionic, ki je tudi predstavjen v tem diplomskem delu. Opisane pa so tudi vrste mobilnega razvoja in razlike med njimi ter slabosti in prednosti posameznih vrst mobilnih aplikacij.

Glavni prispevek aplikacije je omogočiti študentom hitro in enostavno iskanje najbljižjih ponudnikov študentske prehrane. Aplikacija je primerna za vse študente, je pa nenadomestljiva za študente, ki se v določenem okolju ne znajdejo. Na primer študenti, ki so se šele vpisali na fakulteto, pa okolice le te ne poznajo, bodisi tuji študenti, ki pa morda sploh ne poznajo kraja.

Razvita aplikacija za razliko od drugih obstoječih rešitev omogoča, da uporabnik enostavno in hitro ugotovi, kje so najbljižji ponudniki študentske

prehane glede na njegovo trenutno lokacijo. To je glavna funkcionalnost aplikacije, katere odgovor je uporabniku predstavljen takoj ob zagonu aplikacije.

V diplomo je vključena analiza uporabe s pomočjo oblačne storitve Ionic Analytics. V diplomskem delu je predstavljen seznam najbolj priljubljenih restavracij in frekvenca njene uporabe. Pri slednji je težava, saj čas objave aplikacije sovpada s časom začetka poletnih počitnic in je prikaz frekvence uporabe malce pomanjkljiv. V času pisanja zaključka (24.8.2016) si je na platformo Android aplikacijo preneslo 58 uporabnikov, od tega jo 50 uporabnikov še vedno uporablja. Na platformi iOS pa aplikacija še ni bila dosegljiva, saj je bila še v postopku preverjanja s strani distributerja.

5.2 Nadaljnje delo

V spodnjem seznamu so predstavljene dograditve, ki bi jih bilo v prihodnosti smiselno implementirati pri nadaljnjem razvoju aplikacije:

- **Lokalno shranjevanje podatkov o restavracijah**. Trenutno aplikacija ob vsakem zagonu pokliče metodo za pridobitev seznama restavracij. Te podatke bi bilo potrebno hraniti lokalno na mobilni napravi in napisati logiko, ki bi ob vsakem vstopu v aplikacijo preverila starost podatkov in pa povezljivost. Če bi bili podatki mlajši od enega dneva oziroma naprava ne bi mogla dostopati do strežnika, bi se podatki prebrali iz lokalnega pomnilnika, sicer pa pridobili s klicem metode HTTP.

- **Informacija o trenutno odprtih restvracijah**. Trenutno so na zemljevidu vse lokacije restavracij obarvane z rdečo barvo. Ob vstopu v aplikacijo ali na določen časovni interval bi bilo potrebno pregledati, katera restavracija je trenutno odprta. Na podlagi tega bi lahko lokacije odprtih restavracij obrvali z zeleno barvo.

- **Možnost izbire med skokom na podrobnosti restavracije**. Če je na določeni lokaciji več restavracij, se ob kliku na odprto okence odprejo podrobnosti restavracije, ki je na vrhu seznam teh restavracij.

Uporabniku bi bilo potrebno zagotoviti možnost, da izbere restavracijo, katere podrobnosti si želi ogledati.

- **Zmogljivejši iskalnik**. Trenutni iskalnik išče samo po nazivih restavracij. Potrebno bi bilo zagotoviti, da lahko uporabniki poiščejo restavracije tudi po drugih kriterijih, kot naprimer iskanje restavracije z določeno dodatno ponudbo.

- **Priporočanje restavracij glede na število klikov**. Za vsako restavracijo bi lahko dodali še število klikov oziroma obiskov strani s podrobnim opisom restavracije, kar bi lahko uporabnikom služilo tudi kot orientacija o priljubljenosti posamezne restavracije.

Literatura

[1] Raymond K. Camden. *Apache Cordova in Action*. Manning Publications, 2015.

[2] Jeremy Wilken, David Aden, Jason Aden. *AngularJS in Action*. Manning Publications, 2015.

[3] Jeremy Wilken. *Ionic in Action*. Manning Publications, 2015.

[4] HyperText Markup Language. [Online]. Dosegljivo: https://en.wikipedia.org/wiki/HTML. [Dostopano 4. 8. 2016].

[5] JavaScript. [Online]. Dosegljivo: https://en.wikipedia.org/wiki/JavaScript. [Dostopano 18. 8. 2016].

[6] Number of apps available in leading app stores as of June 2016. [Online]. Dosegljivo: http://www.statista.com/statistics/276623/number-of-apps-available-in-leading-app-stores. [Dostopano 15. 8. 2016].

[7] ECMAScript. [Online]. Dosegljivo: https://github.com/tc39/ecma262. [Dostopano 18. 8. 2016].

[8] Cascading Style Sheets. [Online]. Dosegljivo: https://en.wikipedia.org/wiki/Cascading_Style_Sheets. [Dostopano 18. 8. 2016].

[9] Ionic. [Online]. Dosegljivo: http://ionicframework.com. [Dostopano 18. 8. 2016].

[10] Ionic (mobile app framework). [Online]. Dosegljivo: https://en.wikipedia.org/wiki/Ionic_(mobile_app_framework). [Dostopano 18. 8. 2016].

[11] AngularJS. [Online]. Dosegljivo: https://en.wikipedia.org/wiki/AngularJS. [Dostopano 18. 8. 2016].

[12] Apache Cordova. [Online]. Dosegljivo: https://en.wikipedia.org/wiki/Apache_Cordova. [Dostopano 18. 8. 2016].

[13] LoopBack. [Online]. Dosegljivo: https://docs.strongloop.com/display/public/LB/LoopBack. [Dostopano 18. 8. 2016].

[14] MongoDB. [Online]. Dosegljivo: https://en.wikipedia.org/wiki/MongoDB. [Dostopano 18. 8. 2016].

[15] Heroku. [Online]. Dosegljivo: https://en.wikipedia.org/wiki/Heroku. [Dostopano 18. 8. 2016].

[16] Bonar. [Online]. Dosegljivo: https://github.com/BloomSN/bonar. [Dostopano 18. 8. 2016].

[17] Open Data Slovenija. [Online]. Dosegljivo: https://opendata.si. [Dostopano 18. 8. 2016].

[18] Modulecounts. [Online]. Dosegljivo: http://www.modulecounts.com. [Dostopano 19. 8. 2016].

[19] Npm (software). [Online]. Dosegljivo: https://en.wikipedia.org/wiki/Npm_(software). [Dostopano 19. 8. 2016].

[20] ŠOUPit!. [Online]. Dosegljivo: https://play.google.com/store/apps/details?id=si.soup.SOUPit. [Dostopano 22. 8. 2016].

[21] mobileVŠ. [Online]. Dosegljivo: https://play.google.com/store/apps/details?id=si.tovarnaidej.evs. [Dostopano 22. 8. 2016].

[22] ehrana.si. [Online]. Dosegljivo: https://play.google.com/store/apps/details?id=com.trilogic.ehrana. [Dostopano 22. 8. 2016].

[23] Apple App Store. [Online]. Dosegljivo: https://itunes.apple.com/si/genre/mac/id39?mt=12. [Dostopano 22. 8. 2016].

[24] Bonar. [Online]. Dosegljivo: http://www.bonar.si. [Dostopano 22. 8. 2016].

[25] Študentska prehrana. [Online]. Dosegljivo: https://www.studentska-prehrana.si. [Dostopano 22. 8. 2016].

[26] Architectural overview of Cordova platform. [Online]. Dosegljivo: https://cordova.apache.org/docs/en/latest/guide/overview. [Dostopano 22. 8. 2016].